

SRM space manager workflow description

Dmitry Litvintsev *

Timur Perelmutov †

Fermi National Accelerator Laboratory

January 22, 2008

1 Preface

Review of the existing code and db schema uncovered that each new reservation takes more and more time to make as there is dynamic calculation of available space. E.g: when file is put into space reservation we check sum of sizes of all existing files in this reservation, compare it to reservation size and only then put the file in. Same is true for space reservations within linkgroup. In addition it was uncovered that we did not calculate available space for space reservation correctly. We subtracted from `freespaceinbytes` `sizeinbytes` of all spacereservations in a given linkgroup. We should have subtracted `sizeinbytes-usedspaceinbytes` as `freespaceinbytes` already has `usedspaceinbytes` subtracted (this is space taken by files in the linkgroup).

2 Reserve Space

It is suggested to add column `reservedspaceinbytes` to `srmlinkgroup` table. This column holds the sum of sizes of all *unused space* in reservations made in this linkgroup. It can be filled by `before insert` database trigger in `srmsspace` table, or on `SELECT * FROM SRMLINKGROUP FOR UPDATE`.

```
select srmlinkgroup.id,
       coalesce(sum(srmsspace.sizeinbytes-srmsspace.usedspaceinbytes),0)
  from srmlinkgroup left outer join srmsspace on
    srmlinkgroup.id=srmsspace.linkGroupId group by srmlinkgroup.id;
```

A query to update this field (e.g. during switchover to new schema, or some periodic thread) might look like this:

* e-mail: litvinse@fnal.gov

† e-mail: timur@fnal.gov

```

update srmlinkgroup set reservedspaceinbytes=
    select coalesce(sum(s.sizeinbytes-s.usedspaceinbytes),0)
  from srmlinkgroup lg left outer join srmspace s on
    s.linkGroupId=lg.id and lg.id=srmlinkgroup.id);

```

The available linkgroup id then is chosen using a simple query that does not involve table scans or aggregates:

```

SELECT lg.id as id,
lg.freespaceinbytes-lg.reservedspaceinbytes as available
from srmlinkgroup lg, srmlinkgroupvos lgvo
where lg.id=lgvo.linkGroupId
and lg.onlineAllowed = 1
and lg.custodialAllowed = 1
and lg.lastUpdateTime >= 1197403109246
AND ( lgvo.VOGroup = 'testers'
      OR lgvo.VOGroup = '*' )
AND ( lgvo.VORole = 'testers'
      OR lgvo.VORole = '*' )
where lg.freespaceinbytes-lg.reservedspaceinbytes>.... order by available de

```

The meaning of `srmspace.usedspaceinbytes` is discussed below. This field holds sum of file sizes for files in state 2.

2.1 Create Space Reservation

1. select available link group using last query from previous section. If linkgroup id is not found - rollback transaction (SQL issued is `SELECT ... FOR UPDATE`)
2. Else we make entry in `srmspace` table. An example of space reservation of 10000000 bytes:

```

insert into
srmspace values
(328503 , 'testers' , '' , 0 , 1 , 53988 , 10000000000 ,
1197581438295 , 31536000000 , 'Hello_Dolly3' , 0 , 0)

```

3. increment `srmlinkgroup.reservedspaceinbytes` :

```

update srmlinkgroup set
reservedspaceinbytes=reservedspaceinbytes+10000000000

```

4. Commit Transaction

As the result of this actions we have new entry in `srmspace` table and update in `srmlinkgroup` where `srmlinkgroup.reservedspaceinbytes` got incremented.

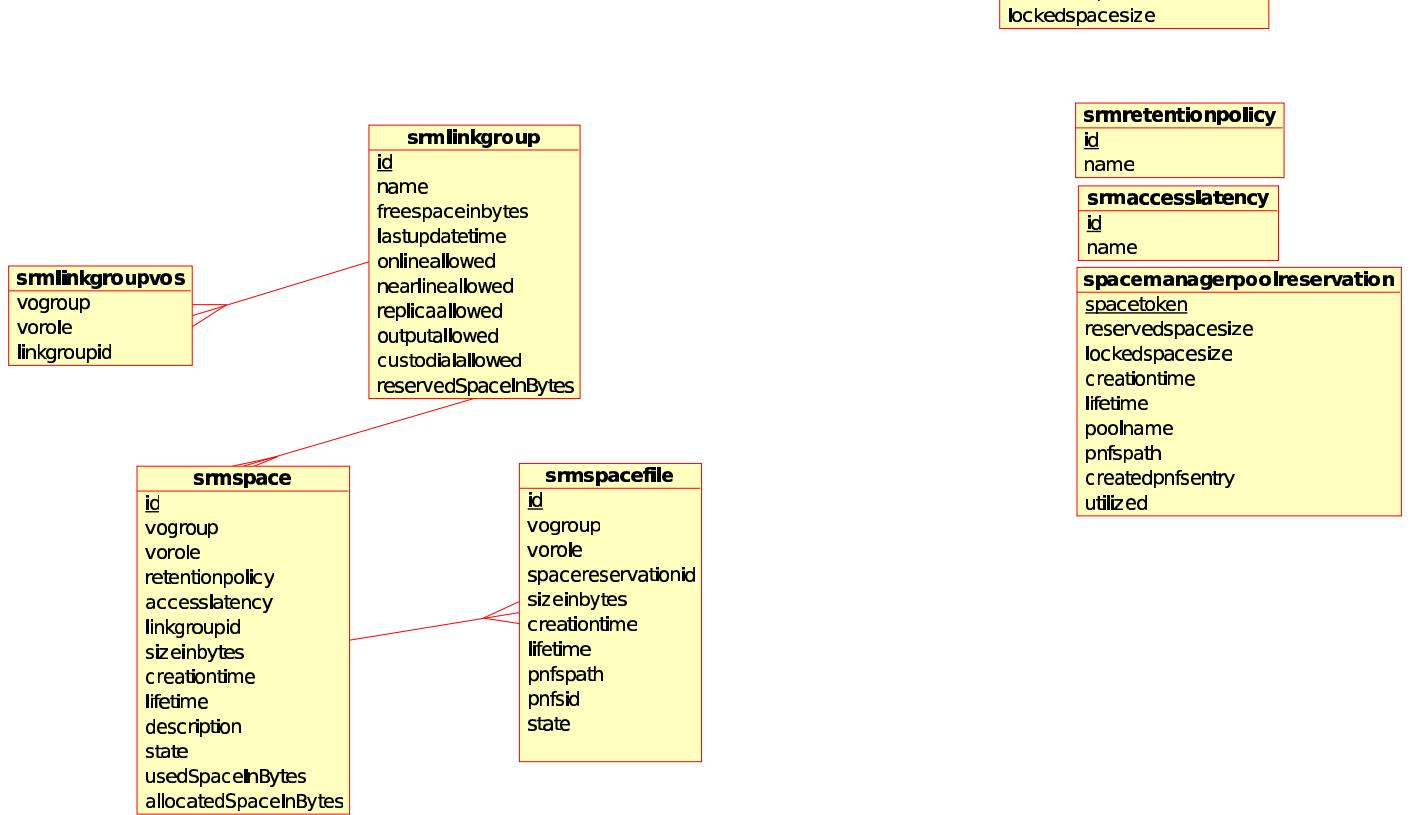


Diagram: entity relationship diagram Page 1

Figure 1: SRM space reservation database schema with two added columns in **srmlinkgroup** and **srmspace** tables

3 Putting file into Space

We added similar two columns `usedSpaceInBytes` and `allocatedSpaceInBytes` to **srmspace** table. These columns will keep cache of sums of sizes of files in state STORED and RESERVED (or TRANSFERRING) correspondingly.

Put file into space is performed by invocation for example:

```
srmcp -2 -space_token=328503 file:///data/litvinse svn/dCacheBuild/big.data \
srm://fapl110.fnal.gov:8443/srm/managerv2?SFN=/pnfs/fnal.gov/data/testers/NULL/litvinse/bigger
```

When the file is put into space the following happens on the server side:

1. lookup for space reservation based on space token. We do

```
select * from srmspace where srmspace.id=... for update
```

2. we check that

`srmspace.sizeinbytes-(srmspace.usedSpaceInBytes+srmspace.allocatedSpaceInBytes)` is greater than (or equal) to the size of the file. If not, rollback transaction. Abort copy. Else we make entry in **srmspacefile**:

```

insert into
srmspacefile values
(328504 , 'testers' , '' , 328503 , 927985664 , 1197581513959 , 3599656 ,
'/pnfs/fNAL.gov/data/testers/NULL/litvinse/bigger2.data' , NULL , 0)

```

3. If file in state STORED we execute update of `srmspace.srmpspace.usedSpaceInBytes`. field incrementing it by the size of the file.

```

update srmspace set
usedspaceinbytes=usedspaceinbites+3599656

```

or if file state is RESERVED (or TRANSFERRING) we increment `srmpspace.allocatedSpaceInBytes`.

```

update srmspace set
allocatedspaceinbytes=allocatedspaceinbytes+3599656

```

4. and (if file in state STORED) decrement `srmlinkgroup.reservedspaceinbytes`

```

update srmlinkgroup set
reservedspaceinbytes=reservedspaceinbytes -3599656

```

Useful queries:

```

select srmspace.id ,
coalesce(sum(srmspacefile.sizeinbytes),0)
from srmspace left outer join srmspacefile on
srmspace.id=srmspacefile.spacereservationid and
srmspacefile.state=2 group by srmspace.id;

```

```

update srmspace set usedspaceinbytes=
(select coalesce(sum(sf.sizeinbytes),0)
from srmspace s left outer join srmspacefile sf on
s.id=sf.spacereservationid and sf.state=2 and s.id=srmspace.id);

```

```

select srmspace.id ,
coalesce(sum(srmspacefile.sizeinbytes),0)
from srmspace left outer join srmspacefile on
srmspace.id=srmspacefile.spacereservationid and
srmspacefile.state<2 group by srmspace.id;

```

```

update srmspace set allocatedSpaceInBytes=
(select coalesce(sum(sf.sizeinbytes),0)
from srmspace s left outer join srmspacefile sf on
s.id=sf.spacereservationid and sf.state<2 and s.id=srmspace.id);

```

4 File Lifecycle

A file in SRM exists in the following 4 states:

```
public static final FileState RESERVED      =
            new FileState("Reserved",      0);
public static final FileState TRANSFERRING =
            new FileState("Transferring", 1);
public static final FileState STORED        =
            new FileState("Stored",       2);
public static final FileState FLUSHED       =
            new FileState("Flushed",     3);
```

When file changes state from “Stored” to “Flushed” we need to update `srmspace.usedSpaceInBytes` decrementing it by the file size and incrementing `srmlinkgroup.reservedspaceinbytes` by the same amount.

When file gets from tape back to the dCache (I do not know into what state it goes) but we need to update `srmspace.usedSpaceInBytes` by incerementing it by the file size amd decrement `srmlinkgroup.reservedspaceinbytes` by the same amount.

I understand that all changes to cached data happen only if the file reaches state=2

4.1 File is deleted

Removal accomplished by executing `srmrm`:

```
srmrm -2 srm://fapl110.fnal.gov:8443/srm/managerv2?SFN=/pnfs/fnal.gov/data/testers/NULL/litvi
```

Naturally we need to find space reserbvation associated with this file, decrement `srmspace.usedSpaceInByt` and then remove the entry associated wiith this file from `srmspacefile`.

1. find `srmspacefile` record:

```
SELECT * FROM srmspacefile WHERE
pnfsId = '00010000000000000796B98' FOR UPDATE;
```

2. remove the record

```
DELETE FROM srmspacefile WHERE id =328504
```

3. update record in `srmspace`

```
SELECT * FROM srmspace WHERE id = 328503 FOR UPDATE
```

```
UPDATE srmspace SET
srmspace.usedspaceinbytes=srmspace.usedspaceinbytes-file_size
```

4. update record in `srmlinkgroup`

```
SELECT * FROM srmlinkgoup where id = space.linkgroupid FOR UPDATE
```

```
UPDATE srmlinkgroup SET  
reservedspaceinbytes=reservedspaceinbytes+file_size
```

5. commit transaction.